Computer Programming II

Levels: 10-12
Units of Credit: 1.00
CIP Code: 11.0202
Core Code: 35020000040
Prerequisites: Algebra I, Keyboarding Proficiency, Computer Technology, Computer Programming IA & IB
Skill Test: 803 Computer Programming II (C++)
835 Computer Programming II (Java)
AP Computer Science

COURSE DESCRIPTION
This is an advanced course in computer programming/software engineering and applications. It reviews and builds on the concepts introduced in CP I. It Introduces students to dynamic allocation of data, advanced utilization of classes, advanced GUI techniques, and advanced applications of recursion through the application of mathematical concepts.

CORE STANDARDS, OBJECTIVES, AND INDICATORS

STANDARD 1
Students will develop applications which make advanced use of the skills and concepts developed in Computer Programming IA & IB.

Objective 1: Demonstrate the ability to develop advanced applications.
a. Develop advanced applications using input, calculations, and output.
b. Develop advanced applications using control structures.
c. Develop advanced applications in object-oriented programming.
d. Develop advanced applications using data structures
e. Develop advanced applications using files (sequential files).

Objective 2: Utilize recursive algorithms
a. Analyze and solve recursive methods
b. Utilize recursive algorithms to solve a problem
c. Identify the base case, recursive case, and action of each recursive function
d. Understand the use of a recursive helper function

Objective 3: Create advanced functions
a. Understand and create overloaded methods.
b. Create and use overloaded operators (C++).

STANDARD 2
Students will use searching and sorting algorithms.

Objective 1: Demonstrate the ability to search data structures in programs.
a. Develop a binary search.
b. Compare efficiency of sequential and binary searches.

Objective 2: Demonstrate the ability to sort data structures in programs.
a. Sort arrays using the selection sort algorithm.
b. Sort arrays using another sorting algorithm (insertion, merge, quick, heap, bubble).
c. Compare the efficiency of differing sorting algorithms.

STANDARD 3
Students will utilize multidimensional arrays.

Objective 2: Utilize multidimensional arrays.
a. Initialize arrays.
b. Input data into arrays.
c. Output data from arrays.
d. Perform operations on arrays.
e. Perform searches on arrays.

STANDARD 4
Students will properly employ dynamic data structures and abstract data types (ADTs).

Objective 1: Demonstrate the ability to use stacks in programs.
a. Declare stack structures.
b. Initialize stacks.
c. Check for empty and full stacks.
d. Push on to and pop off values from stacks.
e. Develop an application that utilizes stacks.

Objective 2: Demonstrate the ability to use queues in programs.
a. Declare queue structures.
b. Check for empty and full queues.
c. Initialize queues.
d. Enqueue values on to and dequeue values off of queues.
e. Develop an application that utilize queues.

Objective 3: Demonstrate the ability to use maps.
a. Define a hashing algorithm and what makes a good hash.
b. Understand collisions and how they are handled.
c. Declare a map structure.
d. Determine whether a key exists in the map.
e. Assign a value to a given key in the map.
f. Retrieve values using a key.
g. Develop an application that utilizes maps.

Objective 4: Utilize type-safe data structures (generics or templates)
a. Utilize data structures that are type-safe using generics or templates.
b. Understand how type-safe data structures prevent errors.
c. Optional -- Demonstrate how to create a template or generic class.

STANDARD 5
Students will design and implement advanced objected oriented concepts.

Objective 1: Implement object oriented programs
a. Create classes that have high cohesion and low coupling.
b. Understand and use composition and aggregation (HAS-A) relationships.
c. Understand the use of class variables (static variables).

Objective 2: Implement Inheritance in an objected oriented program.
a. Utilize class hierarchies (parent-child relationships).
b. Understand IS-A Relationships.
c. Override methods. Understand how to call the overriding method from the child.
d. Understand the protected modifier
e. Call a parent class constructor from the child's constructor

Objective 3: Create and use Abstract Classes

a. Create and implement abstract classes
b. Implement interfaces (purely abstract classes).
c. Abstract classes

Objective 4: Implement polymorphism

a. Demonstrate the use of abstracting a child instance as a parent instance.
b. Determine IS-A relationships at run-time.

Objective 5: Demonstrate overloading techniques.

a. Demonstrate method overloading
b. Demonstrate operator overloading (C++ only)

STANDARD 6

Students will use Unified Modeling Language (UML) to design object oriented programs

Objective 1: Demonstrate the use of an Activity Diagram.

a. Create an activity diagram to flow-chart how an algorithm works.
b. Translate an activity diagram to code.

Objective 2: Demonstrate the use of a Use Case Diagram.

a. Design a class diagram for the class hierarchy of a program.
b. Translate a class diagram into code.

Objective 3: Show how to use a Sequence Diagram

a. Define how objects communicate via a sequence diagram
b. Show how a class diagram's message relates to a function call

STANDARD 7

Students will develop an individual program of significant complexity and size (300-500 lines) and portfolio of their work.

Objective 1: Create an individual program of significant complexity and size (300-500 lines).

a. Create a UML diagram for the project
b. Organize the project in modular programming.

Objective 2: Compile a portfolio of the individual and group programs developed.

a. Include sample design work
b. Include sample program source code and output

Optional STANDARD 8

Students will participate in a work-based learning experience and/or competition.

Objective 1: Participate in a work-based learning experience.

a. Take a field trip to a software engineering firm
b. Participate in a Job shadow
c. Work an Internship
d. Listen to an Industry guest speaker or posst-secondary guest speaker
e. Intverview with an industry representive
f. Participate in a competition